T7 - MSc Pool

T-POO-700

Security

Bootstrap

1.2

{EPITECH.}

The goal of this Bootstrap is to introduce you to some penetration tools, so that you could diagnose your application and patch its weaknesses.

During this process, you should start all your actions by these three routine questions:

- What is a flaw?
- How do you detect it?
- How do you protect yourself?

In order to prevent possible attacks on your site, you will have to think as a hacker would.

You will go from one flaw to another by testing different vectors of attacks and, if there is a bug, try to find out a way to patch it.

Start by listing the main potential flaws, this may be enough to increase your knowledge throughout your experiences!

Several tools are listed below. Get familiar with them, and use them to review the robustness of your application.

DIRB

DIRB is a web content scanner.

It looks for existing web objects. Its launches dictionary attacks on the web server and analyzes its response. It comes with a set of preconfigured words for easy use, but you can also use custom word lists.

In addition, **DIRB** can sometimes be used as a conventional CGI scanner, but remember that a content scanner is not a vulnerability scanner.

The main purpose of **DIRB** is to enable professional auditing of Web applications. Especially in security related tests.

It covers some gaps not covered by standard web vulnerability scanners. **DIRB** searches for specific web objects that other generic CGI scanners can not search for. It does not search for vulnerabilities or web content that may be vulnerable.

The **DIRB** package contains three tools:

- **dirb** - a web content scanner
- **html2dic** - a word list generator from a web page
- **gendic** - a custom word list generator

The following example shows how one can find routes potentially badly configured and therefore exploitable by a malicious user:

Terminal

```
~/T-POO-700> dirb http://my-website.com/ /usr/share/wordlists/dirb/common.txt

---------
DIRB v2.21
By The Dark Raver
---------

START_TIME: Fri May 16 13:41:45 2014
URL_BASE: http://my-website.com/
WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt

---------

GENERATED WORDS: 4592

-- Scanning URL: http://my-website.com/ --
==> DIRECTORY: http://my-website.com/.svn/
+ http://my-website.com/.svn/entries (CODE:200|SIZE:2726)
+ http://my-website.com/cgi-bin/ (CODE:403|SIZE:1122)
==> DIRECTORY: http://my-website.com/config/
==> DIRECTORY: http://my-website.com/docs/
==> DIRECTORY: http://my-website.com/external/
```

You can test dirb online.
You can also try to use different lists of words suggested by dirb.

NMAP (+ NESSUS/OPENVAS + METASPLOIT)

Nmap is a tool for scanning open ports, identifying hosted services and obtaining information about the operating system of a remote computer.

In order to scan remote computer ports, it uses various scanning techniques that rely on protocols such as TCP, IP, UDP, or ICMP.

Similarly, it relies on responses to specific queries to get a fingerprint of the IP stack, often specific to the system that uses it. It can recognize the version of an operating system and the version of listening services (also called daemons) through this method.

By default, **Nmap** performs a TCP SYN scan.

Terminal

```
~/T-POO-700> nmap google.com

Starting Nmap 7.60 ( https://nmap.org ) at 2027-02-31 14:22 CEST
Nmap scan report for google.com (216.58.204.110)
Host is up (0.0094s latency).
Other addresses for google.com (not scanned): 2a00:1450:4007:80a::200e
rDNS record for 216.58.204.110: par10s28-in-f110.1e100.net
Not shown: 998 filtered ports
PORT     STATE   SERVICE
80/tcp   open    http
443/tcp  open    https

Nmap done: 1 IP address (1 host up) scanned in 17.30 seconds
```

Take extra care to the flags to be used with **Nmap**; the tool is very powerful but is highly dependant on these flags. You may miss some important information!

The purpose of **Nmap** is to give information about the remote system. Nothing more.
When carrying out your audits, you will need other tools to find and exploit potential vulnerabilities on the system.
To find the flaws, tools such as nessus or openVAS are references within the environment.
The exploitation part may be done with tools such as metasploit.

Now to put into practice, try to find some information about this URL:

- which kernel is used by the server?

- which OS is used by the server?
- what are the name and version of the ssh service?
- which version of Apache is used by the server?

SQLMAP

sqlmap is one of the most used SQL injection automation tools.

With a vulnerable http request URL, **sqlmap** can exploit the remote database and retrieve the names of databases, tables, columns, all the data in the tables, and so on.

For a list of options and parameters that can be used with the **sqlmap** command, see the sqlmap documentation.

Imagine that we have a URL with an `id` parameter, such as `http://www.my-website.com/user.php?id=73`.
By adding an unexpected character (such as a quote), we can test the server reaction:
`http://www.my-website.com/user.php?id=73'`.

If this URL returns an uncaught error or responds unexpectedly, it is clear that the application did not escape correctly the quote. So, in this case, this `id` input parameter is vulnerable to SQL injection.
We will use **sqlmap** to automate the exploitation of the sql vulnerability.

The command below checks if the `id` parameter is vulnerable and tests different injections.

```
Terminal
~/T-POO-700> python sqlmap.py -u "http://www.my-website.com/user.php?id=73"
[*] starting at 12:10:33
[12:10:33] [INFO] resuming back-end DBMS 'mysql'
[12:10:34] [INFO] testing connection to the target url
sqlmap identified the following injection points with a total of 0 HTTP(s)
requests:
--
Place: GET
Parameter: id
Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE or HAVING clause
Payload: id=73 AND (SELECT 1489 FROM(SELECT COUNT(*),CONCAT(0x3a73776c3a,(SELECT
(CASE WHEN (1489=1489) THEN 1 ELSE 0 END)),0x3a7a76653a,FLOOR(RAND(0)*2))x FROM
INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)
--
[12:10:37] [INFO] the back-end DBMS is MySQL
web server operating system: FreeBSD
web application technology: Apache 2.2.22
back-end DBMS: MySQL 5
```

Here **sqlmap** found the operating system, the web server and the database used.
Replicating this process, use **sqlmap** to extract the accounts of the clients on the database used here

John-the-Ripper

John the Ripper is a password breaking software, used in particular to test the security of a password.
Install it, create a file containing the following hashes and break them.

```
Terminal
~/T-POO-700> cat passwords.txt
$1$00J1VgvS$fRXpAeu9HG7rvkZqAh7AP/
$1$MsnKl0.f$DKtbqyEIBugt3YyCPuP3G1
$1$owvZPc8M$kbk.Euai7CGlDnxbKd59A1
$1$R.0aA.WT$IJJE4UAbzKV5jcP0DT9g0/
```

What kind of hash is this, what are they made of?

NIKTO

Nikto is a web server scanner.

It performs comprehensive testing to detect various information, including more than 6,700 potentially dangerous files/programs, searches for outdated versions of more than 1,250 services and version-specific issues on more than 270 services.

Play with this tool, starting by answering these questions with basic commands:

- which flag performs verbose output?
- which flag performs XSS injections tests?
- which flag performs all tests except the DoS?
- which plugin looks for information in the robots.txt file?

Download this vm now to find out the most relevant flags to use.

Try to find at least the first three flags, if you feel up to it, and you are far enough ahead, you can also try to find the last two.